

Array of Pointers

Definition - Array which stores a address of variable instead of storing actual value.

Syntax:-

datatype * arr-name [size];

↑ char, int, float, double, Any user defined data type

↑ Name of Array

↑ Number of pointer elements

this indicate that the array elements are pointers.

Int array

int arr[5] = {1, 2, 3, 4, 5};

• char arr[5] = {'A', 'B', 'C', 'D', 'E'};

Example:-

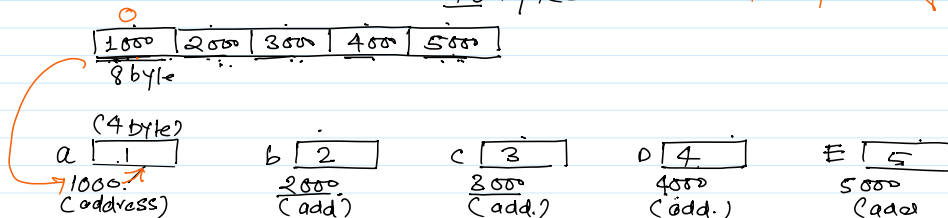
int *arr[5]; // arr is an pointer array which has 5 elements (address of 5 int variable)

Array definition

Memory: 5 pointer = 1 pointer (Byte) * 5
= 8 bytes * 5
= 40 bytes.

pointer takes 8 bytes irrespective of data type.

int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;



arr[0] = &a;
arr[1] = &b;
arr[2] = &c;
arr[3] = &d;
arr[4] = &e;

// Print value of all five variables using pointer array

// Option 1 - Print all manually

printf("%d", *arr[0]) // arr[0] ⇒ 1000

↓ 1000 (address of a)

10 * will dereference pointer & will give the actual value stored at address.

↓

1 (value of a)

printf("%d", *arr[1]); ⇒ 2

printf("%d", *arr[2]); ⇒ 3

printf("%d", *arr[3]); ⇒ 4

printf("%d", *arr[4]); ⇒ 5

// Option 2:- print All variables value using loop :-

for (int i = 0; i < 5; i++) // 0 1 2 3 4

{ printf("%d", *arr[i]); // 1, 2, 3, 4, 5

}

$$i < \text{sizeof(arr)} / \text{sizeof(arr[0])} \Rightarrow i < 40/8 \Rightarrow \underline{i < 5}$$

=x=

Array of char pointers OR char pointers Array

char *arr[5]; // this will store address of 5 char variables.

char a = 'A';
char b = 'B';
char c = 'C';
char d = 'D';
char e = 'E';

1000 2000 3000 4000 5000
arr[0] arr[1] arr[2] arr[3] arr[4]

a 1000 b 2000 c 3000 d 4000 e 5000
1 byte 'A' 'B' 'C' 'D' 'E'

arr[0] = &a;
arr[1] = &b;
arr[2] = &c;
arr[3] = &d;
arr[4] = &e;

for(int i=0; i < sizeof(arr)/sizeof(arr[0]); i++)

{ printf("%c", *arr[i]); } // ABCDE.

}